Factory Pattern – creëren van klassen

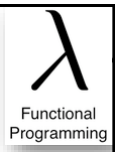
Functional Programming

- **Een eenvoudige pizzafabriek**

Object Oriented Programming

```
public class PizzaFactory {

        public Pizza createPizza(String type) {

            switch (type.toLowerCase())
            {
                case "cheese": return new CheesePizza();
                case "pepperoni":
                        return new PepperoniPizza();
                case "clam": return new ClamPizza();
                case "veggie": return new VeggiePizza();
                default: return null;
            }

        }
}
```

```
import java.util.HashMap;
import java.util.Map;
import java.util.function.Supplier;

public class PizzaFactory {

   private final Map<String, Supplier<Pizza>> factory =
         new HashMap<>();

   public final void add(String type,
                         Supplier<Pizza> supplier)
   {
      factory.put(type, supplier);
   }
```



java.util.function

Interface Supplier<T>

Functional Interface:

This is a functional interface and can therefore be used as the assignment target for a lambda expression or method reference.

```
   public PizzaFactory() {
      add("cheese", CheesePizza::new);
   //add("cheese", () -> new CheesePizza());
      add("pepperoni", PepperoniPizza::new);
      add("clam", ClamPizza::new);
      add("veggie", VeggiePizza::new);    }

   public Pizza createPizza(String type) {
       Supplier<Pizza> supplier =
                     factory.get(type.toLowerCase());
       return supplier!=null ? supplier.get() : null;
   }
}
```